INF

Studiengang
Medien- und
Kommunikationsinformatik

Hochschule Reutlingen
Reutlingen University

Uwe Kloos, Natividad Martínez, Gabriela Tullius (Hrsg.)

Informatics Inside:

Human-Centered Computing

Informatik-Konferenz an der Hochschule Reutlingen
30. April 2014

ISBN 978-3-00-045427-1

Impressum

Anschrift:
Hochschule Reutlingen
Reutlingen University
Fakultät Informatik
Medien- und Kommunikationsinformatik
Alteburgstraße 150
D-72762 Reutlingen

Telefon: +49 7121 / 271-4002
Telefax: +49 7121 / 271-4042

E-Mail: infoinside@reutlingen-university.de
Internet: http://www.infoinside.reutlingen-university.de

Hochschule Reutlingen
Reutlingen University

Herstellung und Verlag: Hochschule Reutlingen
ISBN 978-3-00-045427-1

Inhaltsverzeichnis

Gestenerkennung & Augumented Virtuality

Softwaretechnik

Entwicklung Mobiler Anwendungen

Virtuelle Welten

Optimierung der Usability von digitalen Fahrtenbüchern durch automatisches Erfassen von fahrzeugspezifischen Daten

Viktoria Hoffmann
Reutlingen University
Viktoria.Hoffmann@Student.Reutlingen-University.DE

Abstract

Mit dem starken Wachstum des CarSharing-Angebots und der großen Menge an Flottenfahrzeugen in Unternehmen nimmt auch die Anzahl der Fahrtenbuch-Apps zu. Bei den meisten mobilen Fahrtenbuch-Anwendungen muss der Benutzer den Kilometerstand manuell eintragen. Dies wirkt sich negativ auf die Usability und die User Experience aus. Hinzu kommt, dass jede Minute kostbar ist, die der Fahrer im ausgeliehenen Auto verbringt. Aus diesen Gründen wird hier eine Lösung vorgestellt, bei der der Kilometerstand aus einer Mercedes Benz A-Klasse über den OBD-Anschluss mit Hilfe des CAN Interfaces „ISI b2air“ automatisch ausgelesen und per Bluetooth an die Fahrtenbuch-App der Berger Elektronik GmbH gesendet wird. Hierfür wird mittels der Software „ISI b2app“ die Kommunikation des Diagnosetesters mit dem Fahrzeug aufgezeichnet. Anschließend werden die CAN-Botschaften analysiert und in Bezug auf den Kilometerstand gefiltert. Die entsprechende Anfrage zum Erhalt des Kilometerstandes wird in den Programmcode des Berger Fahrtenbuches implementiert, so dass die App selbstständig den Kilometerstand auslesen kann.

Schlüsselwörter

Usability, User Experience, mobile App, Fahrtenbuch, Automobilindustrie, Fahrzeugtechnik, OBD, CAN, Bussystem, Diagnose, Steuergerät, Kilometerstand auslesen

CR-Kategorien

C.3 [**Special-purpose and application-based systems**]: *Real-time and embedded systems;* J.7 [**Computers in other systems**]: *Real time:* H.5.2 [**User Interfaces**]: *User-centered design*

1 Einleitung

Zurzeit werden von vielen Firmen Fahrtenbücher für Android- und iOS-Smartphones angeboten. Beispiele hierfür sind die Fahrtenbücher von Stefan Meyer[1], Krämer IT[2] oder Ambertation[3].

Betreuer Hochschule: Prof. Dr.-Ing. Natividad Martínez Madrid
Hochschule Reutlingen
Natividad.martinez@Reutlingen-University.de
Betreuer Firma: Dipl.-Ing. Jochen Retter
Berger Elektronik GmbH
Jochen.Retter@bergerelektronik.de

Informatics Inside 2014
Wissenschaftliche Vertiefungskonferenz
30. April 2014, Hochschule Reutlingen

[1] https://itunes.apple.com/de/app/fahrtenbuch/id286070473?mt=8

[2] http://www.androidpit.de/de/android/market/apps/app/de.kraemerit.kfzmobile4/KFZ-Fahrtenbuch-Mobile

[3] https://itunes.apple.com/de/app/drivers-fahrtenbuch/id292648874?mt=8

Mit digitalen Fahrtenbüchern lassen sich Fahrten mit detaillierten Angaben zum Fahrer, Fahrzeug, Fahrtzweck, Kilometerstand, Start und Ziel der Fahrt sowie Zeit und Datum erfassen. Außerdem verfügen viele Fahrtenbücher über eine Backup- und eine Synchronisationsfunktion, um die Fahrten bequem vom PC aus zu verwalten. Fahrtenbücher werden überwiegend in Flottenfahrzeugen und bei CarSharing-Unternehmen für die Dokumentation der Fahrten eingesetzt. Der aktuelle Standort des Fahrzeuges wird bei den meisten Fahrtenbüchern über das GPS des Smartphones automatisch erkannt.

Die Firma TomTom Business Solutions bietet in ihrer Flottenmanagementlösung WORKsmart™ unter anderem auch einen Fahrtenbuchmodus an[4]. Dieser kann bspw. über ein TomTom-Navigationsgerät genutzt werden. Mit Hilfe einer mit dem Fahrzeug verbundenen Blackbox können dann Echtzeitdaten wie Zeit, Ort, Geschwindigkeit und Kilometerstand direkt an das Fahrtenbuch übermittelt werden. Die Anschaffung aller dafür benötigten Soft- und Hardwarekomponenten liegt etwa zwischen 300,- und 400,- €[5]. Die mobile Anwendung eines Fahrtenbuches mit denselben Funktionen ist im Vergleich dazu mit maximal 25,- € deutlich günstiger. Jedoch existiert bislang kein Fahrtenbuch für Smartphones, das direkt mit dem Fahrzeug kommuniziert. Dabei bringt die direkte Kopplung an das Fahrzeug viele Vorteile mit sich. Angaben zur aktuellen Fahrgeschwindigkeit, zum Tachostand, Benzinverbrauch oder Tankfüllung können auf diese Weise automatisiert vom Smartphone ausgelesen werden und das Fahrtenbuch dadurch effektiver und effizienter gestalten. Der Benutzer müsste diese Informationen dann nicht mehr manuell eintragen und würde sich dadurch Zeit – und Geld – beim Bedienen des Fahrtenbuches sparen.

Am Beispiel des Fahrtenbuches der Berger Elektronik GmbH soll gezeigt werden, wie der Kilometerstand eines Fahrzeuges automatisch ausgelesen werden kann. Hierbei werden zwei Komponenten analysiert: zum einen die standardisierte OBD2-Schnittstelle, die den Zugang zu den Steuergeräten gewährt, und zum anderen das Bussystem des Fahrzeuges, das den Datenaustausch zwischen den Steuergeräten regelt. Es werden verschiedene Soft- und Hardwarekomponenten vorgestellt, die dabei helfen, die Buskommunikation aufzuzeichnen, um in Erfahrung zu bringen, mit welcher Botschaft der Kilometerstand angefragt werden kann. Letztendlich soll das Ziel erreicht werden, dass das Berger Fahrtenbuch selbstständig den Kilometerstand ausliest, damit der Benutzer noch effizienter mit dem Fahrtenbuch arbeiten kann.

2 Das Berger Fahrtenbuch

Die Fahrtenbuch-App von der Berger Elektronik GmbH (im weiteren Verlauf Berger Fahrtenbuch genannt, siehe Abbildung 1) ist sowohl für Android- als auch für iOS-Smartphones geeignet.

Abbildung 1: Fahrtenbuch-App der Berger Elektronik GmbH

[4] Vgl. https://de.support.business.tomtom.com/ci/fattach/get/386920/1386661492/redirect/1/session/L2F2LzEvdGltZS8xMzk1MzE2OTM5L3NpZC9sN1NQcElQbA==/filename/ttb-exploreworksmart-int.pdf S.36

[5] http://business.tomtom.com/de_de/products/link/510/highlights/

In der App werden alle relevanten Daten zu den getätigten Fahrten mit einem Fahrzeug erfasst. Dabei handelt es sich um folgende Informationen:

- Fahrer
- Autokennzeichen
- Zeit und Datum
- Start und Ziel der Fahrt
- Kilometerstand
- Fahrtzweck (privat oder geschäftlich, Name der/des Person/Kunden)

3 Fahrtenbuch-Usability

Das Design von mobilen Apps sollte den ISO Normen und Standards für die Gestaltung von interaktiven Systemen entsprechen.

Beim Berger Fahrtenbuch werden einige Daten automatisch von der App erfasst und in die entsprechenden Felder eingetragen. Dabei handelt es sich zum einen um die Zeit und das Datum, die dem Smartphone bekannt sind und zum anderen um den genauen Stand- und Zielort der Fahrt, der jeweils über das GPS-Modul des Smartphones erkannt wird. Allerdings bleiben Daten übrig, die der Anwender selber eintragen muss:

- Fahrer
- Autokennzeichen
- Kilometerstand
- Fahrtzweck (privat oder geschäftlich, Name der/des Person/Kunden)

Den Namen des Fahrers kann der Benutzer entweder über eine hinterlegte Liste auswählen oder selber eintragen. Dasselbe gilt bislang auch für den Fahrtzweck und das Autokennzeichen. Nur den Kilometerstand zu Beginn und Ende einer Fahrt muss der Anwender bisher vollständig manuell eintragen.

Grundsätzlich ist das Eintragen oder Korrigieren vieler Daten von Hand, wie in diesem Fall, für den Benutzer sehr mühsam und zeitaufwendig. Nach Möglichkeit sollte dem Anwender soviel Arbeit wie möglich abgenommen werden. Hierfür muss der Dialog bzw. das interaktive System nach EN ISO 9241-110 (Grundsätze der Dialoggestaltung) gestaltet sein. [1] Speziell das Kriterium der Aufgabenangemessenheit sollte erfüllt sein:

„Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen.“[2]

Bezogen auf das Berger Fahrtenbuch, sollten daher möglichst viele Informationen automatisch von der App eingetragen werden. Da es sich hierbei um fahrzeugspezifische Daten handelt und das Fahrtenbuch während einer Autofahrt verwendet wird, bietet es sich an, den aktuellen Kilometerstand über die Schnittstelle des Fahrzeugs auszulesen und diesen an die App zu senden.

4 Methodik

Um den Kilometerstand aus einem benzin- oder dieselbetriebenen Kraftfahrzeug mit einem Smartphone auslesen zu können, bedarf es einer genauen Analyse der Elektronik im Inneren des Fahrzeugs. Die Steuergeräte (SG) und Bussysteme eines Fahrzeugs kommunizieren durch unterschiedliche – teilweise herstellerspezifische – Protokolle miteinander. Der Kilometerstand ist üblicherweise in allen größeren Steuergeräten gespeichert, z.B. im Motorsteuergerät. Mit einem Diagnosetester, wie er in Werkstätten verwendet wird, kann eine fahrzeugspezifische Diagnose über den genormten On-Board-Diagnose-Stecker (OBD) durchgeführt werden, unter anderem kann dabei auch der Kilometerstand ausgelesen werden. Die Funktion existiert also bereits für Diagnosetester, jedoch nicht für Fahrtenbuch-Apps. Zudem ist der Programmcode des Diagnosetesters durch den Hersteller unzugänglich. Um seine Funktionalität in die Fahrtenbuch-App zu implementieren, gilt es herauszufinden, in welcher Nachricht der Kilometerstand enthalten ist, wie dieser Wert angefordert wird und wie die empfangene Nachricht interpretiert werden muss.

Die Rekonstruktion der Strukturen, Zustände und Verhaltensweisen eines fertigen Systems bzw. das Nachempfinden der Funktionalität dieses Systems wird als „Reverse Engineering“ bezeichnet [3]. Für diesen Zweck soll

ein Versuchsaufbau erfolgen, bei dem die App „ISI b2app“ [4] der Berger Elektronik GmbH als Mithörer zwischen der Kommunikation des Diagnosetesters „mega macs 42 SE“ der Firma Hella Gutmann Solutions GmbH[6] und dem Bussystem einer Mercedes Benz A-Klasse eingesetzt wird.

Die aufgezeichneten Nachrichten werden daraufhin bzgl. des Kilometerstandes analysiert. Die entsprechenden Botschaften zum Anfragen und Erhalten des Kilometerstandes werden im nächsten Schritt in den Programmcode des Berger Fahrtenbuches implementiert. Abschließend wird diese Funktionalität auf verschiedenen Android-Smartphones und Tablets bei entsprechenden Testfahrten in der Mercedes Benz A-Klasse überprüft.

5 Technische Grundlagen der Fahrzeugtechnik

Dieses Kapitel behandelt die interne Kommunikation zwischen den Bussystemen und Steuergeräten in einem modernen Fahrzeug. Dabei wird die Durchführung einer Diagnose beschrieben, der Aufbau eines CAN-Buses und vor allem wie die Nachrichten auf einem Bus gesendet werden.

5.1 On-Board-Diagnose-Stecker

Der On-Board-Diagnose-Stecker (OBD) ist die einzige genormte Hardwareschnittstelle bei Kraftfahrzeugen für externe Geräte. Über diesen 16-poligen OBD-Stecker, der sich üblicherweise im vorderen Fahrer- oder Beifahrerbereich des Fahrzeugs befindet[7] (z.B. im Fußraum oder in der Mittelkonsole), lassen sich verschiedene Daten auslesen, Steuergeräte umkonfigurieren, Fehler diagnostizieren oder neue Software aufspielen.

[6] http://www.hella-gutmann.com/diagnose/mega-macs-42-se/uebersicht/

[7] http://www.obd-2.de/obd-2-stecker-einbauorte.html

Das Gateway im Inneren eines Fahrzeugs dient als Kopplung der verschiedenen Steuergeräte. Wenn vom Diagnosetester eine Anfrage gesendet wird, wird sie über die OBD2-Schnittstelle an das Gateway geschickt und von dort an das entsprechende SG weitergeleitet (siehe Abbildung 2). Im Prinzip arbeitet das Gateway wie ein Router. [5,S.14]

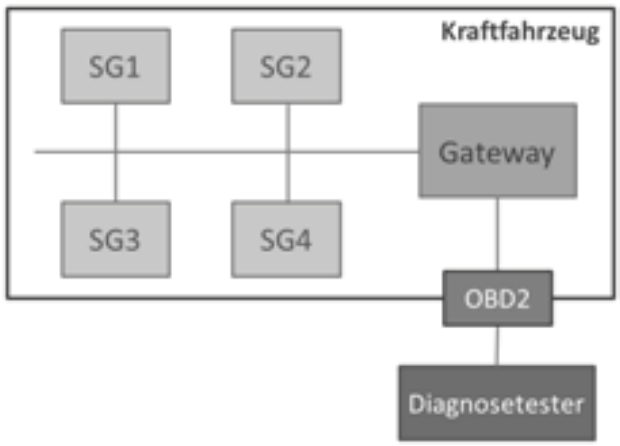

Abbildung 2: OBD2 als Schnittstelle zwischen Diagnosetester und Bussysteme im Kraftfahrzeug (eigene Darstellung i.A.a. [5,S.1])

5.2 Controller-Area-Network

Bussysteme werden als Verbindung zwischen den zahlreich verbauten Steuergeräten in einem Kraftfahrzeug eingesetzt. Das heute am meisten verwendete Kfz-Bussystem ist das Controller-Area-Network (CAN), das von Bosch entwickelt wurde und seit 1991 als internationaler Standard gilt. [5]

Bei der Datenkommunikation richten sich die Bussysteme an das standardisierte ISO/OSI Referenzmodell (siehe Tabelle 1). Der CAN-Bus spezifiziert die Layer 1 (Physical Layer) und 2 (Data Link Layer). Hierbei werden die Bitübertragung, der Botschaftsaufbau und der Buszugriff geregelt. CAN wird bspw. im Motorsteuergerät, im Getriebesteuergerät oder im ABS-Steuergerät verwendet. [5]

Für die gesamte Übertragung von CAN-Botschaften ist ein sog. CAN-Controller zuständig, der jeweils in den entsprechenden Steuergeräten verbaut ist. Eine Botschaft – auch „Frame“ genannt – wird dabei nicht über die Quell- oder Zieladresse identifiziert,

sondern anhand einer eindeutigen Kennzeichnung, dem *Message Identifier*. Beim Senden dieses Frames erhält jedes SG die Botschaft und entscheidet dann mit Hilfe des *Message Identifier*, ob es die CAN-Nachricht verarbeitet oder nicht.

Tabelle 1. ISO/OSI-Referenzmodell [8]

Layer	Bezeichnung	Aufgabe
7	Application	Dienste für den Anwender
6	Presentation	Darstellung der Daten
5	Session	Kommunikationssteuerung
4	Transport	Segmentierung
3	Network	Verbindungssteuerung, Routing
2	Data Link	Botschaftsaufbau, Buszugriff, Fehlersicherung
1	Physical	Bitübertragungsschicht, Elektrische Signalpegel

Insgesamt existieren vier verschiedene Arten von Frames:

- Daten Frame (Übertragung von Nutzdaten bis zu 8 Byte lang)
- Remote Frame (Anfrage eines Daten-Frames)
- Error Frame (signifiziert einen Fehler bei der Übertragung)
- Overload Frame (Pause zwischen Daten und Remote Frame)

Um den Kilometerstand auszulesen, muss das Berger Fahrtenbuch in der Lage sein, CAN-Nachrichten zu senden und zu empfangen. Daher muss sowohl das Anfrage Frame als auch das Daten Frame von der App eingesetzt werden. Die Frames unterscheiden sich zum einen durch das Remote Transmission Bit, das angibt, ob es sich um ein Remote oder Daten Frame handelt und zum anderen durch das Data Field, das die Nutzdaten enthält. Dieses ist beim Remote-Frame leer. Der allgemeine Aufbau einer CAN-Botschaft ist in Abbildung 3 dargestellt. Der ID (Message Identifier) besagt, welcher Wert angefordert wird und der Data Length Code (DLC) gibt Auskunft über die Größe der Nutzdaten. Die anderen Kennfelder des Frames können hier außer Acht gelassen werden. [8]

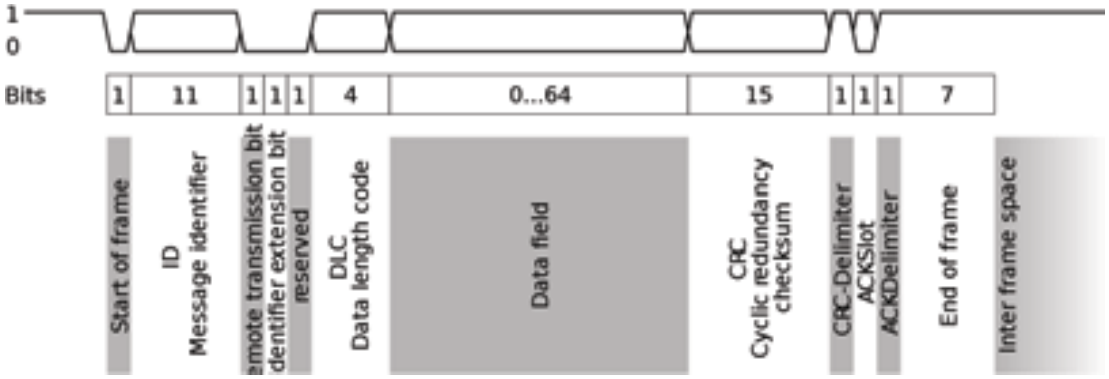

Abbildung 3: Aufbau einer Standard CAN-Nachricht [Quelle: http://upload.wikimedia.org/wikipedia/commons/6/6f/CAN_telegramm_2.0A.svg]

[8] Vgl. K. Reif: Automobilelektronik: Eine Einführung für Ingenieure. Vieweg + Teubner Verlag; 4. Auflage, 2012. S.3f.

6 Verwendete Soft- und Hardware

Neben den Bestandteilen der Fahrzeugtechnik sind noch weitere Soft- und Hardwarekomponenten für die Kommunikation mit dem Fahrzeug notwendig. In diesem Abschnitt werden der Diagnosetester vorgestellt so wie weitere Produkte von Berger Elektronik, die für die Verbindung zum Fahrzeug notwendig sind.

6.1 mega macs 42 SE

Mit dem Diagnosegerät „mega macs 42 SE“ können nicht nur Fehlercodes gelesen und gelöscht, sondern auch digitale Messwerte von Steuergeräten dargestellt werden. Unter anderem kann der aktuelle Kilometerstand des Fahrzeugs ausgelesen werden (siehe Abbildung 4).

Abbildung 4: mega macs 42 SE

Die Verbindung zum Fahrzeug erfolgt drahtlos über Bluetooth. Hierzu muss nur das Diagnosemodul (DT VCI) an den OBD-Stecker angeschlossen werden. [6]

6.2 ISI b2app und ISI b2air

Die Android-Anwendung „ISI b2app“ von Berger Elektronik GmbH ist eine Software für Smartphones und Tablets, mit der die Buskommunikation (CAN, LIN) angezeigt und aufgezeichnet werden kann [4]. Über „ISI b2air“ - ebenfalls von Berger Elektronik - können CAN und LIN Botschaften drahtlos übertragen werden [7]. Dieses Gateway kann die Nachrichten per Bluetooth an einen Laptop, Smartphone oder Tablet senden. Dort können die Nachrichten weiter verarbeitet werden.

6.3 OBD-Breakout-Box

Für die Verbindung zwischen ISI b2air und OBD-Buchse des Fahrzeugs muss ein Kabel eingesetzt werden, dass die richtige PIN-Belegung besitzt, um die Kompatibilität zwischen Fahrzeugschnittstelle und ISI b2air zu gewährleisten. Da sich die PIN-Belegung der OBD-Buchse je nach Automodell teilweise unterscheidet, wird eine OBD-Breakout-Box verwendet. Sie werden zum Beispiel in Laboren oder an Prüfständen eingesetzt, um Kabelverbindungen schnell herzustellen bzw. trennen zu können oder falls kein fertiges Kabel vorhanden ist. Die sogenannten Federstecker werden in die Federbuchsen der OBD-Breakout-Box eingesteckt (siehe Abbildung 5). Auf diese Weise können die 16-OBD-Pins beliebig gesetzt werden. [9]

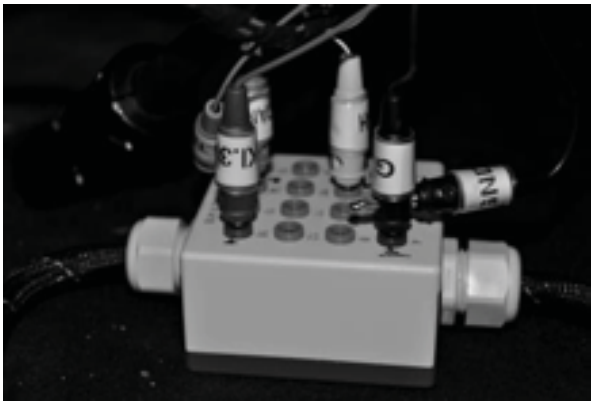

Abbildung 5: OBD-Breakout-Box mit Federsteckern

7 Umsetzung am Beispiel des Berger Fahrtenbuches

Anhand des Berger Fahrtenbuches soll der Kilometerstand aus einer Mercedes Benz A-Klasse automatisch ausgelesen und in das vorgesehene Textfeld eingetragen werden. Nachfolgend wird die Umsetzung dieses Vorhabens beschrieben, für die die Methodik aus Kapitel 4 zum Einsatz kommt.

7.1 Aufzeichnung der Daten und Versuchsaufbau

Um die Rohdaten der Buskommunikation im Fahrzeug zu erhalten, wird zunächst die OBD-Breakout-Box an die OBD-Buchse der Mercedes A-Klasse angeschlossen. Dann werden die Federstecker in die Federbuchsen für Masse, Versorgungsspannung, CAN-High und CAN-Low Datenleitung eingesteckt. Das ausgehende Kabel der Federstecker wird mit dem „ISI b2air“ verbunden. An die OBD-Buchse der OBD-Breakout-Box wird das Diagnosemodul „DT VCI“ angeschlossen. Über Bluetooth verbindet sich dann der Diagnosetester „mega macs 42 SE“ mit diesem Modul. Das letzte Verbindungsstück ist die Android-Software „ISI b2app“, die auch per Bluetooth mit dem „ISI b2air“ verbunden wird. Der gesamte Versuchsaufbau wird in Abbildung 6 skizziert.

Abbildung 7 stellt die Umsetzung dieses Versuchsaufbaus dar. Im Fußraum befindet sich die OBD-Breakout-Box mit dem „ISI b2air“ und dem Diagnosemodul.

Auf dem Fahrersitz ist der Diagnosetester und ein Tablet, auf dem „ISI b2app“ läuft, zu sehen.

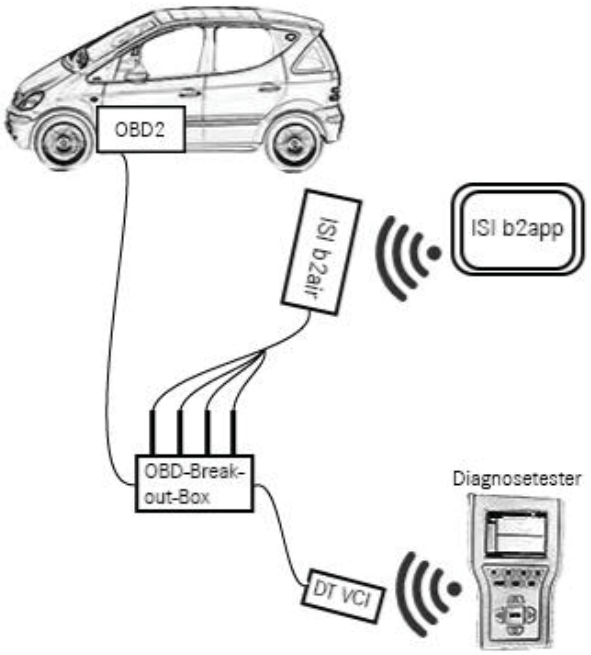

Abbildung 6: Versuchsaufbau als Skizze

Abbildung 7: Versuchsaufbau

Nachdem alle Komponenten miteinander verbunden sind, kann die Aufzeichnung der Daten beginnen. Als aller erstes muss die Zündung gestartet werden (siehe Abbildung 8), damit der Diagnosetester mit dem Bussystem kommunizieren kann. Dann gibt das „ISI b2app“ den Befehl zur Aufzeichnung der Buskommunikation.

Abbildung 8: Prozessablaufdiagramm zur Aufzeichnung der CAN-Nachrichten

Schließlich startet der Diagnosetester mit der Abfrage des Kilometerstandes auf dem CAN-Bus. Das „ISI b2app“ zeichnet währenddessen alle CAN-Botschaften auf, die zwischen dem Diagnosetester und dem Bussystem gesendet werden und speichert sie in einer Logdatei (siehe Abbildung 9).

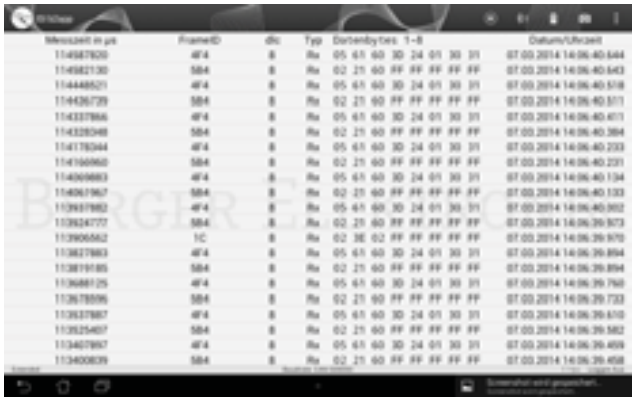

Abbildung 9: aufgezeichnete CAN-Botschaften von ISI b2app

7.2 Analyse der Daten

Aus der Logdatei muss nun das Anfrage Frame und das Daten Frame gefiltert werden, das den Wert des Kilometerstandes anfragt bzw. enthält. Bei den CAN-Daten des Busses werden von Berger Elektronik, so wie allgemein von allen CAN-Tools, nur die relevanten Daten aus dem originalen CAN-Frame-Aufbau (vgl. Abbildung 3) herausgezogen und im Frame dargestellt. Tabelle 2 zeigt einen Ausschnitt solcher CAN-Frames. Zur Ermittlung des Kilometerstandwertes müssen die Datenbytes richtig interpretiert werden. Das Datenbyte 0 entspricht dem Transportprotokoll ISO-TP [vgl. 5, S.119f.]. Es wird zur Übertragung von Botschaften im CAN-Bus verwendet. Gleichzeitig wird mit dem Byte angegeben, wie groß die Nutzdaten sind. Gezählt wird ab dem Datenbyte 1. Das Datenbyte 1 beschreibt das Diagnoseprotokoll KWP 2000 [vgl. 5, S.26f.]. Es gilt als eines der Standardprotokolle zur Kommunikation zwischen einem Diagnosetester und einem SG. Zum Zeitpunkt der Anfrage betrug der Kilometerstand 74783 km. In hexadezimaler Form entspricht das dem Wert `01241F`. Die Speicherung von Zahlenwerten folgt dabei einem bestimmten Format, der sog. Byte-Reihenfolge. Diese wird vom Steuergeräte-Hersteller selber definiert. Im SG der A-Klasse werden die Zahlenwerte nach „Little Endian“ gespeichert [vgl. 5, S.423]. Dies bedeutet, dass das niederwertigste Byte des Wertes an erster Stelle steht. Im Umkehrschluss wird also der HEX-Wert des Kilometerstandes in umgekehrter Reihenfolge abgebildet: an Stelle von `01241F` steht in den Nutzdaten `1F2401`. Dieser Wert entspricht einer Größe von 3 Bytes, d.h. die Nutzdaten des Frames, die den Kilometerstand enthalten, sind bei einer Kodierung ohne Multiplikations- oder Divisionsfaktor 5 Bytes groß. Als weiterer Parameter zur Analyse kann die PID (Parameter ID ~ Datenbyte 2) herangezogen werden [vgl. 5, S.174]. Sie gibt an, welcher Datenwert vom SG angefragt wird. Die PID im Anfrage-Frame wiederholt sich im Antwort-Frame. Daran wird erkannt, dass die beiden Frames zusammengehören. Da die PID-Liste i.d.R. nur der Steuergeräte-Hersteller kennt, kann anhand des Wertes der PID nicht bestimmt werden, ob sich die Anfrage auf den Kilometerstand bezieht.

Tabelle 2. Aufbau der CAN-Nachrichten des Busses der Mercedes A-Klasse

					Datenbytes 1-8 (Data Field)							
Messzeit	**ID**	**DR**	**Typ**	**DLC**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
544.722864	5B4	Rx	d	8	30	08	28	FF	FF	FF	FF	FF
544.769981	4F4	Rx	d	8	21	5F	00	00	05	52	00	00
544.821274	4F4	Rx	d	8	22	32	34	34	35	30	30	31
544.941634	1C	Rx	d	8	02	3E	02	FF	FF	FF	FF	FF
544.968649	5B4	Rx	d	8	02	21	60	FF	FF	FF	FF	FF
544.979916	4F4	Rx	d	8	05	61	60	1F	24	01	30	31
545.087441	5B4	Rx	d	8	02	21	60	FF	FF	FF	FF	FF
545.099853	4F4	Rx	d	8	05	61	60	1F	24	01	30	31

Daher werden die Hinweise zur Größe der Nutzdaten und der HEX-Wert des Kilometerstandes als Hinweise zur Analyse der CAN-Botschaften verwendet. Es werden zunächst alle Frames gefiltert, bei denen im Datenbyte 0 der Wert `05` steht. Viele der Frames enthalten als Nutzdaten die Bytes `1F 24 01`. Daraus lässt sich schließen, dass es sich dabei um den HEX-Wert des Kilometerstandes in umgekehrter Reihenfolge und damit um den gesuchten Datenframe handelt. Der dazugehörige Anfrage-Frame unterscheidet sich immer dadurch, dass der Wert des Datenbyte 1 um hexadezimal 40 kleiner ist. Zum Daten-Frame mit dem Datenbyte-1-Wert `61` gehört demzufolge der Anfrage Frame mit dem Datenbyte-1-Wert `21`.

Der Daten Frame, der den Wert des Kilometerstandes enthält, lautet also wie folgt:

```
4F4 Rx d 8 05 61 60 1F 24 01 30 31
```

Der dazugehörige Anfrage Frame, der die Anfrage beinhaltet, hat folgenden Aufbau:

```
5B4 Rx d 8 02 21 60 FF FF FF FF FF
```

Das Senden des Anfrage Frames reicht aus, um die Antwort zu erhalten. Es muss vorher kein Verbindungsaufbau zum Bussystem hergestellt werden. [5]

Bevor aber die Anfrage in das Fahrtenbuch implementiert werden kann, muss sichergestellt werden, dass eine proprietäre Anwendung nur mit Hilfe des „ISI b2air“ CAN-Nachrichten senden und empfangen kann. Um das zu Testen, wurde der Anfrage Frame zunächst von „ISI b2app“ über das „ISI b2air“ an den CAN-Bus derselben Mercedes A-Klasse gesendet. Dieser Test verlief erfolgreich, denn als Antwort kam der oben erwähnte Daten Frame.

7.3 Implementierung

In den Programmcode des Fahrtenbuchs muss die Funktionalität implementiert werden, die den CAN-Anfrage-Frame sendet, den CAN-Daten-Frame richtig interpretiert und den entsprechenden Kilometerstand in das Textfeld einträgt. Hierfür wird auf bereits vorhandene Softwarebibliotheken für das Senden und Empfangen von CAN-Daten über die Bluetooth-Schnittstelle zurückgegriffen.

Die App wird um weitere Funktionen ergänzt, die den Anfrage Frame an den CAN-Bus senden, die Antwort interpretieren und den Kilometerstand an die entsprechende Stelle schreiben.

Die Programmiersprache ist Java und genutzt wird das frei verfügbare Android SDK[9].

[9] http://developer.android.com/sdk/index.html

Als Entwicklungsumgebung wird Eclipse[10] verwendet.

Folgende Klassen und Methoden wurden in das Berger Fahrtenbuch implementiert:

Klasse `km_handler():`

Der Anfrage Frame wird in einen Byte-Array geschrieben:

```
byte[] message =
{0x02,0x21,0x60,(byte) 0xff,(byte)
0xff,(byte) 0xff,(byte) 0xff,(byte)
0xff};
```

Das Byte-Array `message` wird zusammen mit dem ID `5B4` an den CAN-Bus gesendet. Ein Timer sorgt dafür, dass diese Nachricht jede Sekunde verschickt wird. Sobald eine neue Fahrt angelegt wird, wird die Anfrage gesendet.

Methode `gotCanMsg():`

Diese Methode wird immer dann aufgerufen, wenn das eingesetzte Smartphone eine CAN-Nachricht erhält. Die Botschaft wird von der Methode `gotCanMsg()` analysiert, indem es den ID überprüft. Wenn der Wert des ID `4F4` beträgt, so handelt es sich um das CAN-Frame mit dem Kilometerstand:

```
else if (id == 0x4F4) {
     long l = 0;
     l |= 0xff & data[3];
     l |= ((0xff & data[4]) << 8);
     l |= ((0xff & data[5]) << 16);
     gotKilometerstand((int)(l));
}
```

Der Wert des Kilometerstandes befindet sich im 3., 4. und 5. Datenbyte. Diese HEX-Bytes werden durch bitweise Operation in Dezimalzahlen umgewandelt. Das Ergebnis impliziert den Wert des Kilometerstandes. Dieser wird im letzten Schritt in das entsprechende Textfeld geschrieben und immer aktualisiert, sobald sich der Kilometerstand ändert.

[10] https://www.eclipse.org/downloads/

7.4 Test und Ergebnisse

Die erste Testfahrt in der Mercedes Benz A-Klasse erfolgte mit dem „ISI b2air“, der OBD-Breakout-Box und dem Smartphone LG Nexus 5 mit Android-Version 4.4.2. Zunächst wurde die OBD-Breakout-Box mit den Federsteckern an die OBD-Buchse des Autos angeschlossen. An das andere Ende der OBD-Breakout-Box wurde das „ISI b2air“ angeschlossen. Nach dem Start der Zündung konnte sich das Berger Fahrtenbuch über Bluetooth zum „ISI b2air“ verbinden. Somit konnte im Fahrtenbuch eine neue Fahrt angelegt werden. Dabei müssen zunächst ein paar Informationen vom Benutzer eingetragen werden (vgl. Kapitel 2). Unmittelbar nach dem Anlegen einer neuen Fahrt wurde der Kilometerstand von der App richtig erkannt und in das entsprechende Textfeld automatisch eingetragen. Einzugeben war nur noch der Zweck der Fahrt. Auch während der Fahrt sollte der veränderte Kilometerstand angezeigt werden. Dieser Test verlief ebenfalls erfolgreich. Es wurde stets der richtige Wert dargestellt.

7.4.1 Optimierung

Da die OBD-Breakout-Box mit den Federsteckern und das „ISI b2air“ zusammen viel Platz im Fußraum des Fahrzeugs einnehmen, stört diese Vorrichtung den Fahrer beim Betätigen der Pedale und stellt damit eine gefährliche Situation beim Fahren dar. Unter diesem Risiko darf das Fahrtenbuch unter keinen Umständen benutzt werden. Daher gilt es, diese Vorrichtung so zu optimieren, dass für den Fahrer und eventuelle Beifahrer keine Gefahr im Straßenverkehr besteht.

Auf Grund dessen wurde die OBD-Breakout-Box mit den Federsteckern durch ein „OBD2 Standardkabel DSUB-9“ ersetzt (siehe Abbildung 10). Bei diesem Kabel musste zunächst die PIN-Belegung so angepasst werden, dass die Weiterleitung der CAN-Botschaften vom „ISI b2air“ zum Auto gewährleistet ist. Dafür wurde in der Entwicklungsabteilung bei Berger Elektronik die Verdrahtung der Masse, der Versorgungsspannung, der CAN-

High- und CAN-Low-Leitungen im OBD-Stecker des Kabels umgelötet.

Abbildung 10: OBD-Standard-Kabel DSUB-9

Mit dem Ergebnis, das in Abbildung 11 zu sehen ist, lässt sich wesentlich sicherer und ungestört fahren. Praktischerweise bietet sich das Ablagefach links neben dem Lenkrad gut an, um das „ISI b2air“ zu verstauen.

Abbildung 11: Das umgelötete OBD-ISIb2air-Kabel

7.4.2 Testfahrten

Zur Überprüfung des neuen Kabels wurde ein kurzer Test mit dem Google Nexus 7 durchgeführt, bei dem der Kilometerstand zu Beginn der Fahrt ausgelesen werden sollte. Nachdem dieser Test erfolgreich verlief, wurden drei Testfahrten unternommen, um den gesamten Aufgabenbereich der Fahrtenbuch-App zu prüfen. Hierbei wurden drei Android-Geräte verwendet:

- Google Nexus 5 (Android 4.4.2)
- Samsung Galaxy Nexus S (Android 4.1.2)
- Google Nexus 7 (Android 4.4.2)

Das Fahrtenbuch auf allen drei Geräten sollte den Kilometerstand von Anfang bis zum Ende der Fahrt durchgehend automatisch erkennen und in das vorgesehene Textfeld eintragen. Mit allen drei Android-Geräten wurde ein positives Ergebnis erzielt. Abbildung 12 zeigt denselben Wert (74813 km) am Tacho des Fahrzeugs und im Berger Fahrtenbuch mit dem Smartphone Samsung Galaxy Nexus S.

Abbildung 12: Anzeige des Kilometerstandes am Tacho (links) und im Berger Fahrtenbuch (rechts)

8 Fazit

Mit dem positiven Ergebnis des automatischen Auslesens des Kilometerstandes konnte gezeigt werden, dass auch die Schnittstelle eines Fahrzeugs genutzt werden kann, um die Usability und damit auch die User Experience eines Fahrtenbuches zu verbessern. Diese Methodik kann bei allen mobilen Apps, die in irgendeiner Weise im Straßenverkehr genutzt werden, angewendet werden.

Allerdings kann das Berger Fahrtenbuch nicht bei allen Fahrzeugen den Kilometerstand auslesen. Die entsprechende CAN-Anfrage ist bei jedem Fahrzeug anders. Sie variiert von Modell zu Modell und von Serie zu Serie. Um das Produkt erfolgreich auf den Markt zu bringen, müsste bei einer ganzen Flotte von Fahrzeugen die Buskommunikation analysiert und die entsprechende Botschaft in das Berger Fahrtenbuch implementiert werden.

9 Literaturverzeichnis

[1] M. Herczeg: Software-Ergonomie: Theorien, Modelle und Kriterien für gebrauchstaugliche interaktive Computersysteme. Oldenbourg Wissenschaftsverlag, 2009. S.169ff.

[2] HTW Chur: DIN EN ISO 9241-110 http://www.cheval-lab.ch/was-ist-usability/usability-grundlagen/normen-und-richtlinien/iso-9241-110/ zuletzt geprüft: 08.03.2014.

[3] Springer Gabler Verlag (Herausgeber), Gabler Wirtschaftslexikon, Stichwort: Reverse Engineering: http://wirtschaftslexikon.gabler.de/Archiv/142096/reverse-engineering-v4.html zuletzt geprüft: 24.02.2014.

[4] Berger Elektronik GmbH: ISI b2app http://www.bergerelektronik.com/de/produkte/automotive-smartphone-apps/isi-b2app.html zuletzt geprüft: 08.03.2014

[5] W. Zimmermann, R. Schmidgall: Bussysteme in der Fahrzeugtechnik: Protokolle, Standards und Softwarearchitektur. Vieweg + Teubner Verlag; 4. Auflage, 2011.

[6] Hella Gutmann Solutions GmbH: mega macs 42 SE http://www.hella-gutmann.com/diagnose/mega-macs-42-se/uebersicht/ zuletzt geprüft: 08.03.2014

[7] Berger Elektronik GmbH: ISI b2air http://www.bergerelektronik.com/de/produkte/can/can-bluetooth-wireless-interface-isi-b2air.html zuletzt geprüft: 08.03.2014

[8] T. Dohmke: Bussysteme im Automobil CAN, FlexRay und MOST. Berlin, 2002, S. 4f.: http://de.yu-yongxin.com/klausur/bussysteme.pdf zuletzt geprüft: 14.03.2014

[9] Steve V. Hatch: Computerized Engine Controls. Cengage Learning, 2011, S. 163f.